State Elimination in Accelerated Multiagent Reinforcement Learning

Ary Setijadi Prihatmanto, Widyawardana Adiprawita, Safreni Candra Sari, and Kuspriyanto

Department of Electrical Engineering, School of Electrical Engineering and Informatics
Bandung Institute of Technology, Jalan Ganesha 10, Bandung, 40132, Indonesia
asetijadi@lskk.ee.itb.ac.id, wadiprawita@stei.itb.ac.id,
safrenicsari@yahoo.com, kuspriyanto@lskk.ee.itb.ac.id,

Abstract: This paper presents a novel algorithm of Multiagent Reinforcement Learning called State Elimination in Accelerated Multiagent Reinforcement Learning (SEA-MRL), that successfully produces faster learning without incorporating internal knowledge or human intervention such as reward shaping, transfer learning, parameter tuning, and even heuristics, into the learning system. Since the learning speed is determined among others by the size of the state space where the larger the state space the slower learning might become, reducing the state space can lead to faster convergence. SEA-MRL distinguishes insignificant states of the state space from the significant ones and then eliminating them in early learning episodes, which aggressively reduces the scale of the state space in the following learning episodes. Applying SEA-MRL in gridworld multi robot navigation shows 1.62 times faster in achieving learning convergence. This algorithm is generally applicable for other multiagent task challenges or general multiagent learning with large scale state space, and perfectly applicable with no adjustments for single agent learning situation.

Keywords: Multiagent Reinforcement Learning, multiagent learning, Reinforcement Learning, Accelerated Reinforcement Learning, Machine Learning, state space reduction.

1. Introduction

Multiagent systems are rapidly finding applications in a variety of domains, including robotics, distributed control, telecommunications, and economics [1]. The complexity of many tasks arising in these domains makes them difficult to solve with preprogrammed agent behaviors. The agents must, instead, discover a solution on their own, using learning. A significant part of the research on multiagent learning concerns reinforcement learning (RL) techniques.

In RL, learning is carried out online through trial-and-error interactions of the agent with an environment that provides a reinforcement (reward or penalty) at each interaction. In the particular case of multiagent systems, the reinforcement received by each agent depends both on the dynamics of the environment and on the behavior of other agents, and therefore a multiagent reinforcement learning (MRL) algorithm must address the resulting nonstationary scenarios in which both the environment and other agents are present. Unfortunately, convergence of any RL algorithm requires extensive exploration of the state-action space, which can be very time consuming [2], not to mention the existence of multiple agents also increases the size of the state-action space, therefore, worsening the performance of RL algorithms with respect to convergence (even to suboptimal control policies) when it is adapted to multiagent problems. Therefore acceleration of learning processes is one of important issues in reinforcement learning [3, 4].

Most successes in accelerating MRL incorporated internal knowledge or human intervention into the learning system, such as reward shaping [5-8], transfer learning [6, 9-13], parameter tuning [14], and even heuristics [2, 11, 13, 15]. These approaches could be no longer solutions to RL acceleration where internal knowledge is not available. This paper proposed a novel approach in improving the MRL learning performance called by accelerating the speed of the learning convergence without involving heuristics or any internal knowledge.

Received: February 1st, 2015. Accepted: September 27th, 2016
DOI: 10.15676/ijeei.2016.8.3.12

Since the learning convergence is determined by the size of the state space where the larger the state space the slower learning might become, reducing the state space by eliminating the insignificant states can lead to faster learning. In this paper a novel method called State Elimination in Accelerated Multiagent Reinforcement Learning (SEA-MRL) is presented. This algorithm distinguishes insignificant states from the significant one from early learning episode, which reducing the state space during the learning process. This paper extends our previous work which investigates the use of State Elimination in single agent domain [16].

The remainder of this paper is organized as follows. Section II briefly reviews MRL approaches and describes the multiagent Q learning (MA-Q(λ)) and multiagent SARSA (MA-SARSA(λ)) algorithm, while Section III presents a review of some existing approaches to speed up RL. Next, Section IV shows how the learning speed can be improved by eliminating some insignificant states from the state space during the learning process. Then, section V details the mapping gridworld multirobot navigation into SEA-MRL and the experiments performed in the domain, and analyses the results obtained. Finally, section VI presents conclusions and future directions.

2. Multiagent Reinforcement Learning

As the framework for MRL In this paper, we adopt the Markov Games (MGs) Theory. An MG is an extension of a Markov decision process (MDP) that uses elements from game theory and allows the modeling of systems where multiple agents compete among themselves to accomplish their tasks.

Formally, an MG [17] over a set of n agents is defined as:

1) S: a finite set of environment states.
2) $A_1, \dots, A_n$: a collection of finite sets A_i, with the possible actions for each agent i, $1 \leq i \leq n$.
3) $T: S \times A_1 \times \dots \times A_n \times S \rightarrow [0,1]$: a state transition function where $T(s, a_i, \dots, a_n, s')$ defines the probability of a transition from state s to state s' when the agents execute respective actions $a_1, \dots, a_n$.
4) $R_i: S \times A_1 \times \dots \times A_n \rightarrow \mathbb{R}$: a reward function associated to each agent *i*.

   The goal of the i^{th} agent is to find a policy $\pi_i: S \rightarrow A_i$ that maximizes the expected sum of discounted rewards, $E\{\sum_{j=0}^{\infty} \gamma^j r_{i,t+j}\}$, where $r_{i,t+j}$ is the reward received j steps in the future by agent *i*, and $\gamma \in [0,1]$ is the discount factor.

In this paper we adapt the fully cooperative SG as a framework for MRL. These heterogeneous agents with differing action spaces learn to achieve the overall system goals by implicitly cooperating to achieve their common goal: the global reward function. Together the agents form a multiagent system (MAS). Each agent's state space consists of the full state space as would be used in single agent solutions, but does not include any information on action selection done by the other agents. Claus and Boutilier (1998) call this approach Independent Learners (IL). During policy evaluation, each agent selects its own action, without any form of negotiation or central coordination.

Busoniu et al. [18] proposed the cooperative setting of Multiagent Reinforcement Learning with Independent Learners [19] case , where the action-value function of an agent *i* builds upon the basic Q-learning algorithm [20], which is given by:

$$Q_{t+i}^i(s_t^i, a_t^i) = (1 - \alpha_t^i) Q_t^i(s_t^i, a_t^i) + \alpha_t^i [r_{t+1}^i + \gamma \max_{a^i \in A^i} Q_t^i\,(s_{t+1}^i, a^i)], \tag{1}$$

where $s_t^i \in S^i$ is the state of agent *i* at time step *t* and all other agent-specific terms have been similarly superscripted by *i*. Another cooperative setting for MRL with IL case adopted in this paper is based on SARSA which is given in the following equation,

$$Q_{t+i}^i(s_t^i, a_t^i) = (1 - \alpha_t^i) Q_t^i(s_t^i, a_t^i) + \alpha_t^i [r_{t+1}^i + \gamma\, Q_t^i\,(s_{t+1}^i, a^i)], \tag{2}$$

and the utility function is

$$V^i(s) = \max_{a^i \in A^i} Q^i(s^i, a^i) \tag{3}$$

where the agent's policy is now a probability distribution over actions, $\pi(s) \in P(A)$, and $\pi(s, a)$ is the probability of taking action a. The optimal policy $\pi *(s)$ is the one that produces the largest V for every state s.

$$\pi^{i*}(s) = \text{argmax}_{a^i \in A^i} Q^{i*}(s^i, a^i) \quad \forall s \in S. \tag{4}$$

A widely used action selection policy that includes exploratory actions is the ϵ-greedy policy $\pi_{\epsilon-greedy}(s_t, a_t)$ which is defined such that a random action is selected with probability ϵ (uniformly sampled from A) and $a_{t,greedy}$ otherwise:

$$\pi_{\epsilon-greedy}(s_t, a_t) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|}, & if\ a_t = a_{t,greedy} \\ \frac{\epsilon}{|A(s)|}, & if\ a_t \neq a_{t,greedy} \end{cases} \tag{5}$$

with $\epsilon \in [0,1]$ the exploration rate and $|A(s)|$ the number of actions in A within state s. For a good trade-off between exploration and exploitation, the value for ϵ is typically chosen from the range [0.01,0.20] [21].

3. Approach on Accelerated Multiagent Reinforcement Learning

The quality of the learning itself is measured based on eventual convergence to optimal, speed of convergence to optimality and regret [4]. Although many algorithms come with a provable guarantee of asymptotic convergence to optimal behavior [7], an agent that quickly reaches a plateau at 99% of optimality may, in many applications, be preferable to an agent that has a guarantee of eventual optimality but a sluggish early learning rate. Therefore the speed of convergence to near-optimality is more practical to be measured. The speed of convergences to the near optimality with high dimension environment is often big issues in RL. One effort that can be applied to accelerate RL is to find a new algorithm that reduces the state space by carefully eliminating some unimportant states while learning. If not careful enough then the potentially important state might also be eliminated, and the learning process will fail.

Several methods have been proposed to speed up RL. One of them is incorporate the prior knowledge into RL. Mataric [8] used implicit domain knowledge to design the reinforcement/reward function in situated domains based on utilizing heterogeneous reward functions and goal specific progress estimator. Laud and De Jong [9] formulated an explanation of the potential of reward shaping to accelerate reinforcement learning with a reward-based analysis. Konidaris and Barto [10] introduced the use of learned shaping rewards in RL tasks, where an agent uses prior experience on a sequence of tasks to learn a portable predictor that estimates intermediate rewards, resulting in accelerated learning in later tasks that are related but distinct. Matignon, Laurent et al. [11] accelerate goal-directed RL by modifying the reward function using a binary reward function (for discrete state space) and continuous reward function (for continuous state space) and implementing Gaussian goal biased function as the initial values of *Q(s)*. Ma, Xu et al. [12] applied a state-chain sequential feedback Q-learning algorithm for path planning of autonomous mobile robots in unknown static environments, where the state chain is built during the searching process.

Another approach in accelerating the RL is by applying transfer learning in RL. The core idea of transfer is that experience gained in learning to perform one task can help improve learning performance in a related, but different, task [13]. Drummond [14] used transfer learning from the related tasks, which generate a partitioning of the state space which is then used to index and compose functions stored in a case base to form a close approximation to the solution of the new task. Taylor and Stone [15] introduced behavior transfer, a novel approach to speeding up traditional RL. Celiberto, Matsuura et al. [2] applied transfer learning from one agent to another agent by means of the heuristic function speeds up the convergence of the algorithm. Case-based is used to transfer the learning, and it makes TL-HAQL algorithm. Peters and Schaal [16] reduced the problem of learning with immediate rewards to a reward-weighted regression problem with an adaptive, integrated reward transformation for faster convergence. Takano, Takase et al. [17] accelerated the learning process by implementing the effective transfer learning method, which merges a selected source policy to the target policy without negative transfers. Norouzzadeh, Busoniu et al. [18] used two transfer criteria in measuring agent's performance (by the distance between its current solution and the optimal one and by the empirical return obtained) to decide when to transfer learning from an easier task to a more difficult one so that the total learning time is reduces.

More recent proposal in accelerating RL is to include heuristics in RL algorithms. Gao and Toni [19] incorporate heuristic, represented by arguments in value-based argumentation into RL by using Heuristically Accelerated RL techniques in RoboCup Soccer Keepaway-Takeaway game. Celiberto, Matsuura et al. 2010 [2] applied transfer learning from one agent to another agent by means of the heuristic function speeds up the convergence of the algorithm. Case Based (CB) is used to transfer the learning, and it makes RL algorithm faster. Terashima, Takano et al. [20] used the prior information on the problem utilizing options as prior information. In order to increase the learning speed even with wrong options, methods for option correction by forgetting the policy and extending initiation sets. Bianchi et al. [21] presented a novel class of algorithms, called Heuristically-Accelerated Multi-agent Reinforcement Learning (HAMRL), which allows the use of heuristics to speed up well-known multi-agent reinforcement learning algorithms. Such HAMRL algorithms are characterized by a heuristic function, which suggests the selection of particular actions over others.

Other approaches were also proposed. Senda, Mano et al. [22] reduced state space by modelling the state space by 3D space coordinates where then the space model is simplified by converting 3D coordinates to 2D coordinates under a certain terms. Grounds and Kudenko [23] investigated the use of parallelization in RL, with the goal of learning optimal policies for single-agent RL problems more quickly by using parallel hardware. Braga and Araújo [24] influenced zone algorithm, an improvement over the topological RL agent (TRLA) strategy, that allows reducing the number of requested interactions, which is based on the topological-preserving characteristic of the mapping between states (or state–action pairs) and value estimates. Kartoun, Stern et al. [25] allowed several learning agents to acquire knowledge from each other. Acquiring knowledge learnt by an agent via collaboration with another agent. Price and Boutilier 2003 [26] proposed an implicit imitation that can accelerate reinforcement learning dramatically in certain cases, roughly by observing a mentor, a reinforcement learning agent can extract information about its own capabilities in, and the relative value of, unvisited parts of the state space. Potapov and Ali [27] tuned the learning steps, discount and exploration degree parameters to influence the convergence rate. McGovern, Sutton et al. [28] used built in policies or macro-actions as a form of domain knowledge that can improve the speed and scaling of reinforcement learning algorithms.

The algorithm proposed in this paper is aimed to improve the RL learning performance by accelerating the speed of the learning convergence without involving heuristics or any learning domain prior knowledge. Since the learning convergence is determined by the size of the state space, where the larger the state space the slower learning might become, reducing the state space can lead to faster learning. Instead of heuristics or any learning domain prior knowledge, this proposed method identifies some potential consistent local minima states to be considered as insignificant states and is considered to be eliminated from the state space. This method reduces the state space, decreasing the computation order and hence accelerating the convergence speed.

4. Online State Elimination to Accelerate Reinforcement Learning

The complexity of an algorithm is often expressed using big O notation. Big O notation is useful when analyzing algorithms for efficiency. In RL, if a good task representation or suitable initialization is chosen, the worst-case complexity of reaching a goal state has a tight bound of $O(n^3)$ action executions for Q-learning and $O(n^2)$ action executions for value-iteration [32], where n stands for number of states in the state space. If the agent has initial knowledge of the topology of the state space or the state space has additional properties, the $O(n^3)$ bound can be decreased further. In our case, where prior knowledge is not available, initial knowledge is not incorporated in the new algorithm.

Since the worst case complexity depends totally on number of states, it's very clear that n has very dominant factor in determining the convergence speed, where reducing n will lead to decreasing the computation needed to reach learning convergence. When robot learns to master a new skill, it learns to determine which states considered important to support its performance. Robot learns to classify which states are significant, and which states are less significant. By updating its *Q(s,a)* values every iteration, agent update its policies by choosing the highest Q value as its decision factor, which means it starts to ignore smaller Q value (which indicates less significant states). This condition forces the agent to rarely visit these less significant states until agent succeeded in maximizing its rewards.

Almost all RL algorithms are based on estimating value functions--functions of states (or of state-action pairs) that estimate how good it is for the agent to be in a given state (or how good it is to perform a given action in a given state). The notion of "how good" here is defined in terms of future rewards that can be expected, or, to be precise, in terms of expected return. Of course the rewards the agent can expect to receive in the future depend on what actions it will take. Accordingly, value functions are defined with respect to particular policies. The value of the state space in this case represents the significance factor of the state. High value state represents the high probability that agent will decide in determining its optimum policy π^*. A high value state means significant states that have to be maintained in the state space because it provides solution to the agent. On the other hand the states that have less value become less interesting for the agents. The probability to visit these states is towards 0 in 100% exploitation cases. When the insignificant factor of these states can be measured, the states which have high insignificancy can be considered to be eliminated from the state space leaving it reduced.

In order to determine the insignificance rate of a state, in this paper a new function ι is proposed.

Definition 1: The insignificance function $\iota : S \longrightarrow \mathbb{R}$ is a function that returns a value indicates the insignificance rate of a state $s \in S$.

This insignificance function represents an insignificance rate of a state that derived from value *V(s),* where the lowest value V of the neighborhood state is considered to be potentially insignificant. This function indicates which state $s \in S$ is insignificant enough that it can be ignored and should be eliminated from the state space, since they don't provide solutions to the agent.

Definition 2: If the domain X is a metric space then f is said to have a local (or relative) maximum point at the point x^* if there exists some $\varepsilon > 0$ such that $f(x^*) \geq f(x)$ for all x in X within distance ε of x^*. Similarly, the function has a local minimum point at x^* if $f(x^*) \leq f(x)$ for all x in X within distance ε of x^*.

Definition 3: A state is called a local minimum state at k^{th} iteration or s_{min}^{k} when its V value is proven to be a local minimum of all *V(s)* function in k^{th} iteration for all $s \in S$.

Since the first learning iteration every state in state space of agent i accordingly has initial insignificance value $\iota^i(s) = 0$ for all $s^i \in S^i$, as the initial value of the $V^i(s)$. When agent updates its state-action value by harvesting rewards $R^i(s)$ every time it visits a state, insignificance factor $\iota^i(s)$ is also updated by the following return:

$$\iota^i_{k+1}(s) = \begin{cases} \iota^i_k(s) + \mu & if\ s^i\ is\ s^{i,k}_{min} \\ \iota^i_k(s) = 0 & otherwise \end{cases} \tag{6}$$

where $\iota^i_{k+1}(s)$ represents the insignificance value ι of state s^i in the $(k+1)^{th}$ iteration, $\iota^i_k(s)$ the insignificance value of state s^i in k^{th} iteration, and μ is the insignificance step which represents the increasing potential of a insignificant state. The insignificance of state s^i is updated every iteration but it will be reset back to 0 when in the next iteration s^i is no longer a local minimum ($s^i \neq s^{i,k}_{min}$.).

Definition 4: A sub state space $S^{i,k}_{min} \subset S^i$ is a state space of agent *i* at *k*th iteration, which consists of all $s^i \in S^i$ and $\iota^i_k(s) \geq \text{I}$.

When $\iota^i_k(s)$of a state s^i larger than a threshold value I, the state will be added to a sub state space $S^{i,k}_{min} \subset S^i$, which is then considered to be eliminated from the state space.

$$\text{S}^i_{k+1} \longleftarrow \text{S}^i_k \cap S^{i,k}_{min} \tag{7}$$

In every iteration *k*th, agent *i* reduces its Q value to only the new state space S^i_{k+1}, and original action space A_i. However learning at early stages is essentially random exploration {Bianchi, 2013 #127}. Deciding which states is more significant than others in this stage gives very small contributions since every states has its own significance potential. It's very important however to expand I to an exponential function that vary to iteration number k as given in the following equation

$$\text{I}(t) = \text{I}_0 e^{\text{h/k}} \tag{8}$$

where I_0 is a initial value of I and h is a real number. This function gives $\text{I}(t) = \text{I}_0$, when k goes to infinity. This has to be done since in early stage/exploration stage to let agent see all possibilities that it can profit from its *V(s)*. The complete algorithm in pseudo code is given in Figure. 1.

```
Define number of episodes $k$
Define elimination start episode $p$
for all agent $i \in [1 \dots n]$ do
Initialize $Q^i(s,a)$ arbitrarily
$\iota^i(s) = 0$ (for all $s \in S^i$)
k=0
Repeat (for each episode $k$):
        Initialize $s$
        Repeat(for each step of episode):
            Choose $a^i$ from $s$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
            Take action $a^i$, observer $r^i$, $s'$
            $Q^i(s,a^i) \leftarrow Q^i(s,a^i) + \alpha\left[r^i + \gamma \max_{a' \in A_i} Q^i(s',a'^i) - Q^i(s,a^i)\right]$
            $s \leftarrow s'$;
until $s$ is terminal
if $k > p$
            $\text{V}^i(s) = \max_a Q^i(s,a^i)$
            $k=k+1$
            Repeat (for all $s \in S^i$)
                if $\text{V}^i(s)$ is local minimum
                        $\text{I} \longleftarrow \text{I}e^{h/k}$
```

```
                    $\iota^i(s) = \iota^i(s) + \mu$
                    if $\iota^i(s) \geq \mathrm{I}$
                        $S_\iota \leftarrow S_\iota \cup \{s\}$
                    else
                        $\iota^i(s) = 0$
                        $S_\iota \leftarrow S_\iota$
                    Endif
                Endif
                $S^i \leftarrow S^i \backslash S^i_\iota$
                until $s$ is terminal
            end if
        end for
```

Figure 1. SEA-MRL Algorithm

SEA-MRL is applicable for both multiagent and single agent system. Changing the value of number of agents (i) to 1 will automatically convert the environment to single agent environment. For more comperhensive explanation for single agent situation, the reader is referred to [16].

5. Mapping Gridworld Robot Navigation into Multiagent Reinforcement Learning

One of the dominant topics in current mobile robotics research is that of autonomous navigation. In the robot navigation problem, the robots need to find an optimal navigable path in a given environment, with certain constraints imposed on the robot, such as a time limit or limited availability of resources. Optimal path here refers to a path between the two points: the source and destination, which has the least path cost, or in other words the most profitable one among all the existing paths.

The environment is a discrete grid-world with randomly located obstacles. There are three robot agents on the grid-world, starting from an arbitrary initial position. The robot agent or simply called an agent, can occupy a single empty tile at a time and is faced with the task of navigating through the map in an autonomous manner. There can only be one agent on one tile at a time. The agent is capable of sensing its immediate environment and moving in 5 directions (action) one tile at a time respectively North, South, West, East and stay put, that makes the action space $A = \{N, S, W, E, SP\}$ available for the agent. The grid-world environment is given in Figure 2

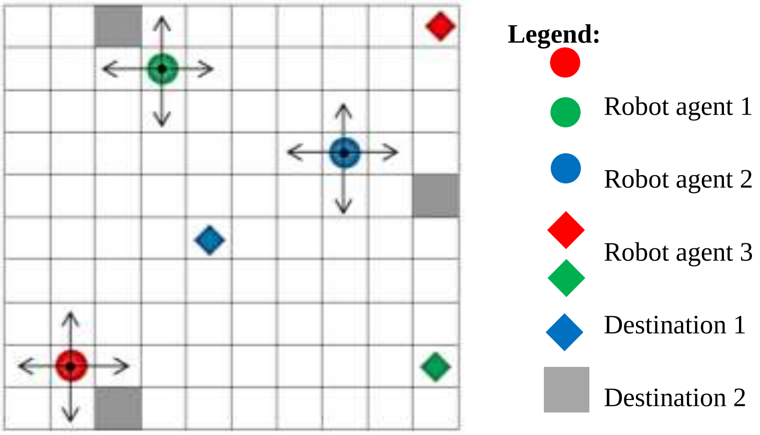


Figure 2. Three robot agents on the 10x10 gridworld can move to 5 directions resp. North, South, West, East, and stay put, learn to find shortest path to its destination and avoiding obstacles.

The state space of agent i or S^i in this environment is defined as:

$$S^i = \{(p_i);\ p_i \in \{(1\ 1), (1\ 2), \dots, (10\ 10)\}\} \qquad (9)$$

where p_i is the position of agent i.

The task of the agent i is to find sequence of actions that have to be performed to achieve the goal state, which is the destination1 for agent 1, destination 2 for agent 2, an destination 3 for agent 3. The global reward function $R(s, a, s')$ for all agents are given as follows

$$R(s, a, s') = \begin{cases} r^{\rightarrow}, s \in S^{\rightarrow} \\ r^{+}, s \in S^{+} \\ r^{-}, s \in S^{-} \end{cases} \qquad (10)$$

where $r^{\rightarrow}$, r^{+}, and r^{-}, is resp. the reward when agent takes action to move to one of the 5 directions, when agent achieved the goal state, and when agent bumped the wall or the obstacle.

Every learning episode starts from the same initial position where agent is always in the same grid. The agent performs the learning task until the episode is ended. There are 3 situations that end the episode: when the agent arrived at the destination state, and when maximum trial had been achieved.

For action selection the following ϵ-*greedy* scheme is used

$$\pi(s, a) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|}, a = arg\ \max_{a' \in A(s)} Q(s', a') \\ \frac{\epsilon}{|A(s)|}, \qquad else \end{cases} \qquad (11)$$

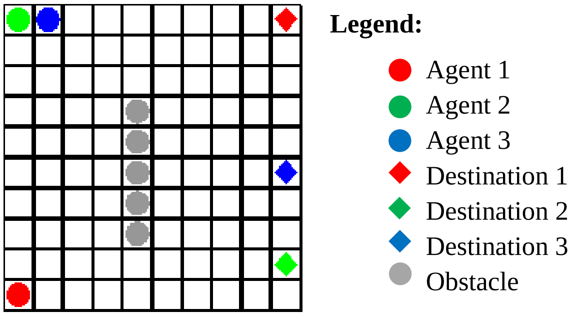
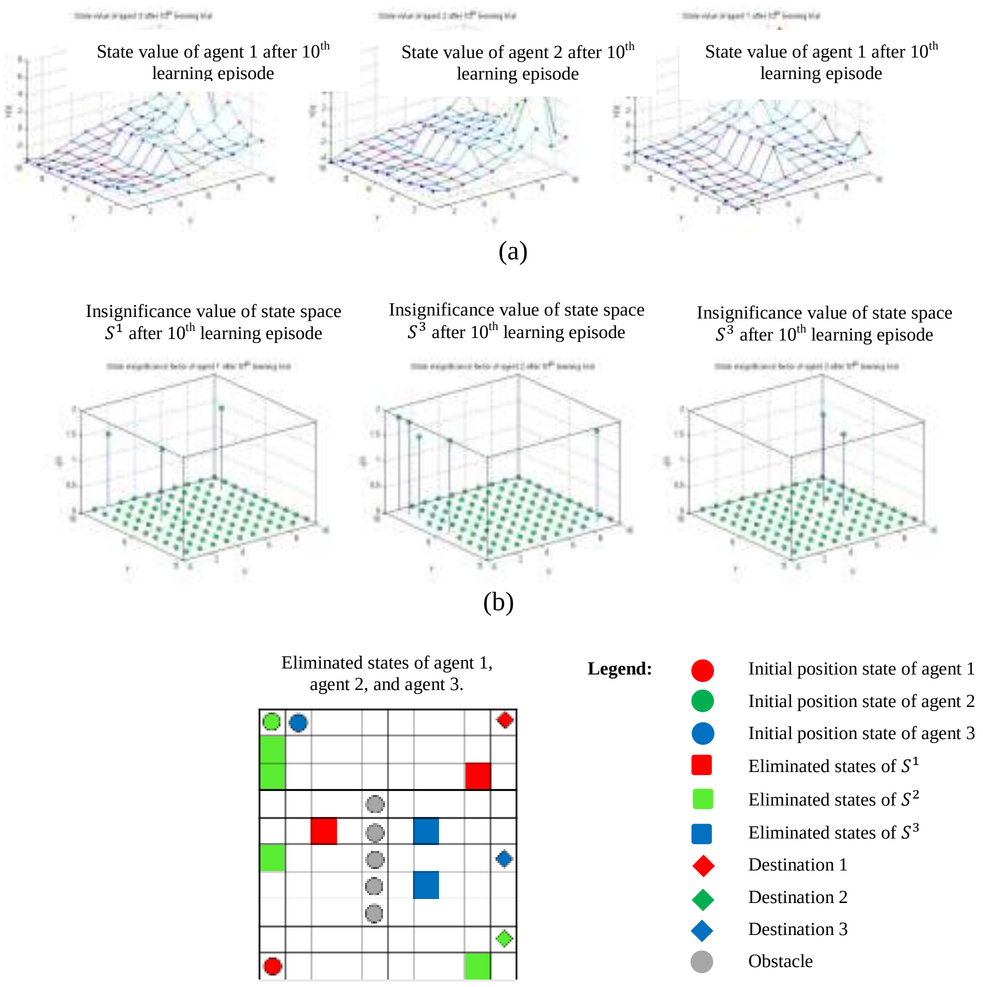


Figure 3. Initial position of agents, obstacles, and destinations on grid world.

In this experiment, the objects initial positions are the same for all algorithms. The position of robot agent 1, agent 2, and agent 3 are resp. on gridworld coordinate (1,1), (1,10), and (2,10) represented by red, green and blue dot as given in Figure. 3. Obstacles were placed on (1,5) and (5,5). Destination 1, 2 and 3 are resp. on (10,10), (10,2), and (10,5).

The agent distinguishes the insignificant states from the significant one starting from the kth episodes when ever $\iota^i(s^i) > \mathrm{I}$, includes them in S_ι^i , and finally eliminates the from the state space S^i. This procedure updates S^i at every learning episode since kth episode. In Figure. 4, the situation of the learning process at k= 10 is shown. State values of agent i, $V^i(s^i)$ is given in (a), while the insignificance function of s^i is given in (b).

Agent i eliminates the local states s^i that has insignificance value $\iota^i(s^i) > \mathrm{I}$. (c) shows the eliminated states, where states given by colour red is eliminated from agent 1's local state space S^1, while colour green by agent 2's local state space S^2, and blue by agent 3's local state space S^3.



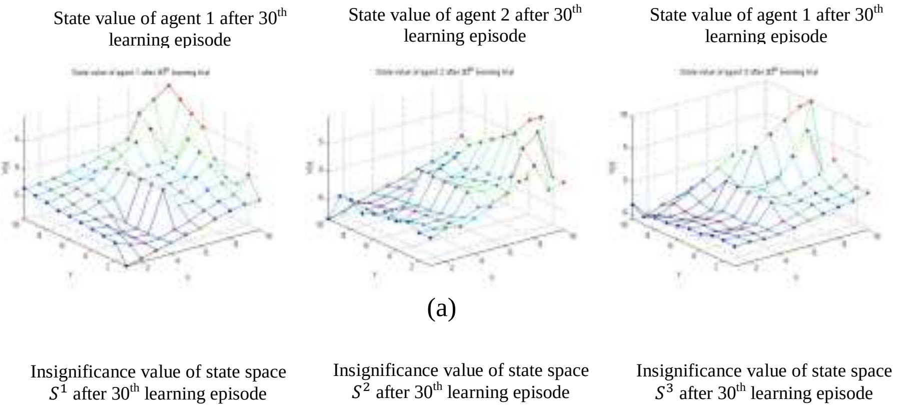
(a)

(b)

(c)

Figure. 4. The situation of the learning process at k= 10.

Agent i start to calculate the insignificance rate ι^i of all states $s^i \in S^i$ while it performs its state exploration.



(a)

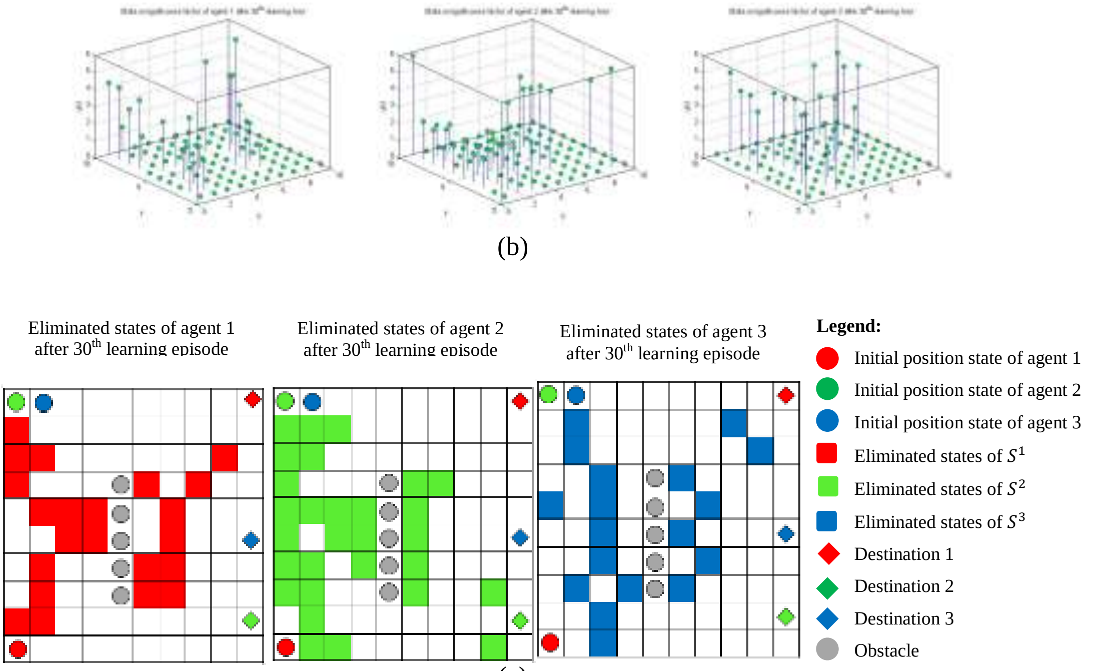


Figure. 5. The growth of the eliminated states in 30th learning episode.

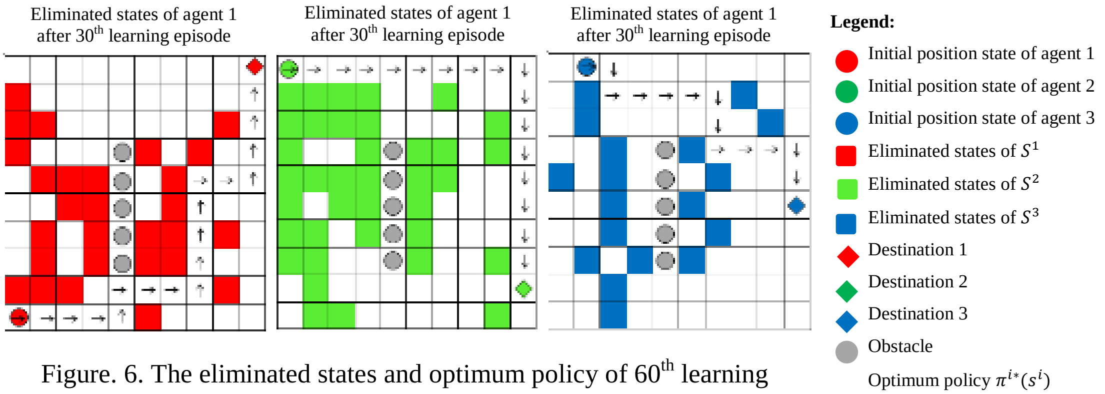

Eliminated states of agent 1 after 30th learning episode
Eliminated states of agent 1 after 30th learning episode
Eliminated states of agent 1 after 30th learning episode
Legend:
Initial position state of agent 1
Initial position state of agent 2
Initial position state of agent 3
Eliminated states of S^1
Eliminated states of S^2
Eliminated states of S^3
Destination 1
Destination 2
Destination 3
Obstacle
Optimum policy $\pi^{i*}(s^i)$

Figure. 6. The eliminated states and optimum policy of 60th learning episodes.

Throughout the learning process, more consistent local minima occur in the state value function. The numbers of eliminated states start to grow. The growth of the eliminated states in 30th episode is shown in Figure. 5. State values of agent i, $V^i(s^i)$ is again given in (a), while the insignificance function of s^i is given in (b). The eliminated states are shown in (c), and finally the optimum policy of agent i, $\pi^{i*}(s^i)$ is given in Figure. 6 together with the final eliminated states at 60th learning episode.

The learning results is displayed in Figure 7, where the experiment was first run using MA-Q(λ) algorithm.The horizontal axis represents the numbers of episodes or trials that was done in the experiment, the vertical axis shows how many steps or iterations needed to complete the task. The learning parameters of the agent are setup as follows: learning rate α=0.3, discount factor γ=0.95, eligibility trace decay factor λ=0.5, and initial exploration probability ϵ=0, decaying with the trials count, the $((r^{\rightarrow}, r^{+}, r^{-}) = (-1, 10, -2)$. The action selection mechanism is given as ϵ-greedy policy $\pi_{\epsilon-greedy}(s_t, a_t)$ as given in eq.16. The agents estimate their own action-value function using eq. (1) for SEA-MRL-Q(λ) and (2) for SEA-MRL-SARSA(λ).

In the episodic learning the number of trials which indicates how many episodes to allow learning to run at most is set to be 100. The maximum iterations to allow a trial to run at most is set to be 500. The algorithms were implemented in Matlab and executed in desktop, with 4GB of RAM in a Windows 7 OS platform.

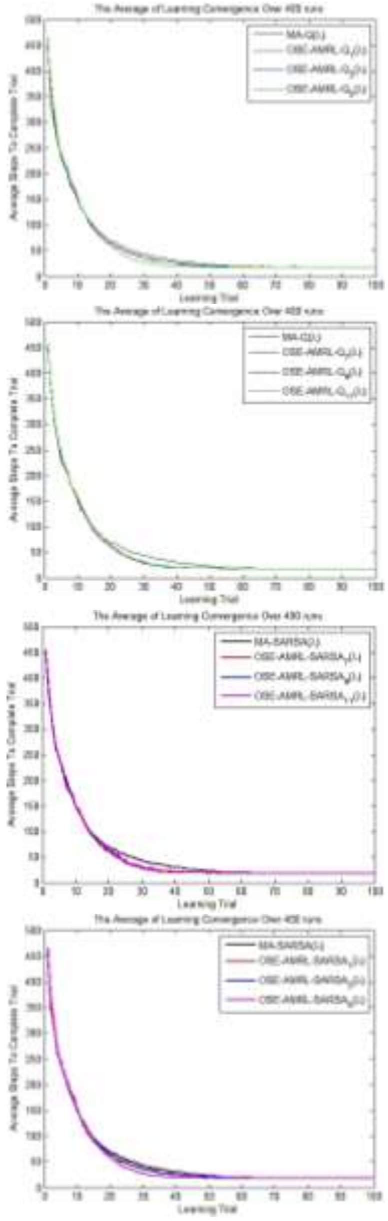


Figure 7.The learning performance of MA-Q(λ) versus SEA-MRL-Q(λ) ((a) and (b)) and MA-SARSA(λ) versus SEA-MRL-SARSA(λ) ((c) and (d)) on 10x10 grid area for robot navigation task. The plot shows the average of the learning convergence over 400 independent runs, where each run consists of runs with 100 trials.

For each experiment, a learning session consisted of 400 runs of 100 trials/episodes each. A trial finished whenever agent had arrived at the goal state or when 500 moves/iteration were completed. Each result is presented as a learning curve derived from the average steps to complete trial, which is the number of moves to achieve goal state. The experiment was conducted as follows. MA-Q(λ), MA-SARSA(λ), SEA-MRL-Q(λ) and SEA-MRL-SARSA(λ) are applied respectively to the robot navigation problem. The performance of each algorithm is plotted and benchmarked. In Figure. 7 the MA-Q(λ) is compared to 6 kinds of SEA-MRL-Q(λ), where SEA-MRL-Q$_p$(λ) represents the performance of SEA-MRL-Q(λ) eliminated from episode $p \in \mathbb{Z}$. In Figure. 7 The MA-SARSA(λ) is also compared to 6 kinds of SEA-MRL-SARSA(λ).

Figure 7 shows the learning curves of the agent averaged over 400 learning sessions. The horizontal axis represents the numbers of episodes or trials that was done in the experiment, while the vertical axis shows how many steps or iterations needed to complete the task. Figure. 7 (a) shows the learning performance curve of the MA-Q(λ), and SEA-MRL-Q(λ) starting at 1st, 3rd, and 5th episodes. Even though the curve of MA-Q(λ) had reached its convergence after 63rd trial, the SEA-MRL-Q(λ) was noticeably better from early episodes due to the use of state elimination method. SEA-MRL-Q$_1$(λ) and SEA-MRL-Q$_3$(λ) reached respectively its convergence already at 50th trial and 40th trial for SEA-MRL-Q$_5$(λ). Similarly in Figure 7 (b) SEA-MRL-Q$_7$(λ), SEA-MRL-Q$_9$(λ) and SEA-MRL-Q$_{11}$(λ) reached its convergence respectively at 40th, 41st, and 42nd trial. Figure 7(c) presents the learning curves of the MA-SARSA(λ), and SEA-MRL-SARSA(λ) starting at 1st, 3rd, and 5th episodes. The results show that the MA-SARSA(λ) reached its convergence after 63rd trial, while SEA-MRL-SARSA$_1$(λ) and SEA-MRL-SARSA$_3$(λ) reached 51st and 50th trial, and SEA-MRL-SARSA$_5$(λ) after 40th trial. Lastly, Figure. 7 (d) presents the learning curves of the SARSA(λ), and SEA-MRL-SARSA(λ) starting at 7st, 9rd, and 11th episodes. The results show that the SEA-MRL-SARSA$_7$(λ), SEA-MRL-SARSA$_9$(λ) and SEA-MRL-SARSA$_{11}$(λ) achieve its convergence respectively after 42nd, 42nd and 45th trial. These results show that by eliminating insignificant states from early learning episodes will speed up the convergence to 1.46 times faster time. The performance analysis of the algorithm is given in Table 1.

RL Algorithm	p	#Successful run	Successful percentage (%) (A)	Convergent after (trials)	Acceleration factor (B)	Overall Learning Performance (A x B)
MA-Q(λ)	-	400	100	63	1	1
SEA-MRL-Q$_p$(λ)	1	244	61	50	1,26	0,7686
	3	360	90	50	1,26	1,134
	5	301	75,25	40	1,58	1,18895
	7	339	84,75	40	1,58	1,33905
	9	366	91,5	41	1,54	1,4091
	11	372	93	42	1,5	1,395
MA-SARSA(λ)	-	400	100	63	1	1
SEA-MRL-SARSA$_p$(λ)	1	264	66	51	1,24	0,8184
	3	358	89,5	50	1,26	1,1277
	5	316	79	40	1,58	1,2482
	7	328	82	39	1,62	1,3284
	9	354	88,5	42	1,5	1,3275
	11	361	90,25	42	1,5	1,35375

Table 1 shows the performance analysis of average (over the 400 learning runs) of the robot agent. From 400 runs, SE-ARL-$Q_p(\lambda)$ starts to reach above 90 % for p > 7. It can be seen that SE accelerate the learning speed up to 1.58 faster for $Q(\lambda)$ and 1.62 for SARSA (λ) (even though there still is failure in learning proses when the state elimination is executed from early stages). The failure happened because some important states were also eliminated, because in early stages, these important states still have low state value. This failure did not happen when agent had chance to explore the states in early episodes, which can be seen on the table that when elimination is started from 9th trial and up, the learning process is above 90% successful.

Table 1. Performance Analysis MA-$Q(\lambda)$ versus SEA-MRL-$Q(\lambda)$ and MA-SARSA(λ) versus SEA-MRL-SARSA(λ).

Finally the learning performance is measured by calculating the ratio of the successful runs and Acceleration factor. All SE-AMRL-$Q(\lambda)$ acceleration factor in the table had outperformed MA-$Q(\lambda)$, and SE-AMRL-SARSA(λ) had outperformed MA-SARSA(λ). The state elimination executed in MA-$Q(\lambda)$ and MA-SARSA(λ) had performed faster convergence speed due to decreasing number of states in the state space.

6. Conclusion and Further Research

Multiagent Reinforcement Learning (MRL) is the solution of multi robot learning problem, since the robot environment is mostly dynamic and stochastic. However the increasing number state and action space leads to problem in MRL since it requires larger memory and computation time which can cause degrading in the learning performance to very poor and even lead to failure in learning.

Since the learning convergence is determined by the size of the state space where the larger the state space the slower learning might become, reducing the state space by eliminating the insignificant states can lead to faster learning. Applying state elimination in gridworld multi-robot navigation had shown significant acceleration factor in multi-agent RL to 1.62 faster convergence speed.

When internal knowledge such as reward shaping, transfer learning, parameter tuning, and even heuristics is no longer applicable to MRL problems, online state elimination becomes a promising solution to accelerate MRL. This algorithm is not only applicable for primitive robot soccer task, but also for other robotic soccer task challenges with large scale state space. This method has also clearly given us a starting point on several promising extension to the existing work and highlighted important new questions. This research has not only resulted in advances in imitation learning, but also has opened up a whole new way of exploring this field that posses an abundant source of ready-to-explore problems for future research.

7. Refferences

[1]. Busoniu, L., R. Babuska, and B. De Schutter, “A Comprehensive Survey of Multiagent Reinforcement Learning. Systems, Man, and Cybernetics”, Part C: *Applications and Reviews, IEEE Transactions on*, 2008. **38**(2): p. 156-172

[2]. Bianchi, R.A.C., et al., “Heuristically-Accelerated Multiagent Reinforcement Learning. Cybernetics”, *IEEE Transactions on*, 2013. **PP**(99): p. 1-1

[3]. Sutton, R.S. and A.G. Barto, “Reinforcement Learning: An Introduction1998”: MIT Press

[4]. Kaelbling, L.P., M.L. Littman, and A.W. Moore, “Reinforcement learning: a survey. J. Artif”. Int. Res., 1996. 4(1): p. 237-285

[5]. Mataric, M.J. “Reward Functions for Accelerated Learning. in ICML”. 1994

[6]. Konidaris, G. and A. Barto. Autonomous shaping: “Knowledge transfer in reinforcement learning”. *in Proceedings of the 23rd international conference on Machine learning*. 2006. ACM

[7]. Matignon, L., G.J. Laurent, and N. Le Fort-Piat, “Reward function and initial values: better choices for accelerated goal-directed reinforcement learning”, in Artificial Neural Networks–ICANN 20062006, Springer. p. 840-849

[8]. Ma, X., et al., “State-chain sequential feedback reinforcement learning for path planning of autonomous mobile robots”. *Journal of Zhejiang University Science* C, 2013. **14**(3): p. 167-178

[9]. Drummond, C., “Accelerating reinforcement learning by composing solutions of automatically identified subtasks”. *Journal of Artificial Intelligence Research (JAIR),* 2002. 16: p. 59-104

[10]. Taylor, M.E. and P. Stone. “Speeding up reinforcement learning with behavior transfer. in AAAI 2004 Fall Symposium on Real-life Reinforcement Learning”. 2004

[11]. Celiberto, L.A., et al. “Using transfer learning to speed-up reinforcement learning”: a cased-based approach. in *Robotics Symposium and Intelligent Robotic Meeting (LARS), 2010 Latin American*. 2010. IEEE

[12]. Norouzzadeh, S., L. Busoniu, and R. “Babuska. Efficient Knowledge Transfer in Shaping Reinforcement Learning”. in *Proceedings of the 18th IFAC World Congress*. 2011

[13]. Takano, T., et al., “Transfer Learning Based on Forbidden Rule Set in Actor-Critic method”. *International journal of innovative computing information and control*, 2011. **7**(5 B): p. 2907-2917

[14]. Potapov, A. and M. Ali, “Convergence of reinforcement learning algorithms and acceleration of learning”. *Physical Review E*, 2003. **67**(2): p. 026706

[15]. Gao, Y. and F. Toni, “Argumentation Accelerated Reinforcement Learning for RoboCup Keepaway-Takeaway”, in *Theory and Applications of Formal Argumentation,* E. Black, S. Modgil, and N. Oren, Editors. 2014, Springer Berlin Heidelberg. p. 79-94

[16]. Sari, S.C., A.S.P. Kuspriyanto, and W. Adiprawita, “Online State Elimination in Accelerated reinforcement Learning”. *International Journal on Electrical Engineering and Informatics*, 2014

[17]. Littman, M.L. “Markov games as a framework for multi-agent reinforcement learning”. in *The Eleventh International Conference on Machine Learning*. 1994. Morgan Kaufmann

[18]. Busoniu, L., B.D. Schutter, and R. Babuska. “Multiagent reinforcement learning with adaptive state focus”. in *17th Belgian-Dutch Conference on Artificial Intelligence (BNAIC-05)*. 2005. Brussels, Belgium

[19]. Claus, C. and C. Boutilier. “The dynamics of reinforcement learning in cooperative multiagent systems”. in *National Conference on Artificial Intelligence (AAAI-98). 1998*

[20]. *Watkins, C.J.C.H. and P. Dayan, Q-Learning. Machine Learning*, 1992. **8**(3-4): p. 279--292

[21]. Schuitema, E., “Reinforcement Learning on Autonomous Humanoid Robots”, 2012

[22]. Laud, A. and G. DeJong. “The influence of reward on the speed of reinforcement learning: An analysis of shaping”. in ICML. 2003

[23]. Taylor, M.E. and P. Stone, “Transfer learning for reinforcement learning domains: A survey. The Journal of Machine Learning Research”, 2009. **10**: p. 1633-1685

[24]. Peters, J. and S. Schaal, “Reinforcement learning by reward-weighted regression for operational space control”, in Proceedings of the 24th international conference on Machine learning2007, ACM: Corvalis, Oregon. p. 745-750

[25]. Terashima, K., H. Takano, and J. Murata, “Acceleration of Reinforcement Learning with Incomplete Prior Information”. JACIII, 2013. **17**(5): p. 721-730

[26]. Senda, K., S. Mano, and S. Fujii. “A reinforcement learning accelerated by state space reduction”. in *SICE 2003 Annual Conference*. 2003. IEEE

[27]. Grounds, M. and D. Kudenko, “Parallel reinforcement learning with linear function approximation”, in *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning* 2008, Springer. p. 60-74

[28]. Braga, A.P.d.S. and A.F.R. Araújo, Influence zones: “A strategy to enhance reinforcement learning. Neurocomputing”, 2006. **70**(1–3): p. 21-34

[29]. Kartoun, U., et al. “Collaborative Q (λ) Reinforcement Learning Algorithm-A Promising Robot Learning Framework”. in *IASTED International Conference on Robotics and Applications (RA 2005),* Cambridge, U.S.A. 2005. ACTA Press

[30]. Price, B. and C. Boutilier, “Accelerating reinforcement learning through implicit imitation”. *J. artif. intell. res.(jair)*, 2003. **19**: p. 569-629

[31]. McGovern, A., R.S. Sutton, and A.H. Fagg. “Roles of macro-actions in accelerating reinforcement learning”. in *Grace Hopper celebration of women in computing*. 1997

[32]. Koenig, S. and R.G. Simmons, “Complexity Analysis of Real-Time Reinforcement Learning Applied to Finding Shortest Paths in Deterministic Domains”, 1992, Carnegie Mellon University

Ary Setijadi Prihatmanto was born in Bandung, Indonesia on August 1972. He received his Bachelor and Master degree in Electrical Engineering from ITB, and doctorate degree in informatics from Johannes Kepler University of Linz. His research interests include dualism of computer vision & computer graphics, human-computer interface, brain-computer interface, game theory on intelligent system and its applications.

Widyawardana Adiprawita, lecturer at STEI-ITB. Received his electrical engineering degree at Electrical Engineering ITB with honour in 1997. Finished master degree at Informatics Engineering ITB in 2000. He finished his doctoral degree with honour at Electrical Engineering ITB. His Research interests are embedded system, robotics and intelligent agent autonomy. He has written more than 20 papers published in international and national publication.

Safreni Candra Sari was born in Medan on January 14, 1975, completed her undergraduate and master study in electrical engineering, Technishe Universiteit Delft the Netherlands in 1999. She followed her second master study in electrical engineering majoring Digital Media and Game Technology at the Institut Teknologi Bandung Indonesia and had completed her study in July 2010. Safreni worked as a faculty member at the General Achmad Yani University (UNJANI) in Bandung, her research interests are robotic system, robotic soccer system, learning in robotics, reinforcement learning, and multi agent learning system.

Kuspriyanto was born in Yogyakarta Indonesia, 2 January 1950, completed his undergraduate degree at Electrical engineering, Institut Teknologi Bandung in 1974. He received his Master and Doctoral degree from Université des Sciences et Techniques de Montpellier (USTL) France. He is currently a Full Professor at the Department of Electrical Engineering, Institut Teknologi Bandung, Indonesia. His current research interests include real time computing systems, computer architecture, and robotics.